AFWL-TR-89-01

(2)

A MODEL OF PARALLEL PERFORMANCE

E. A. Carmona
M. D. Rice

April 1989

AFWL

Final Report

Approved for public release; distribution unlimited.

AD-A207 792

DTIC
ELECTE
MAY 16 1989
S D
H

AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base, NM 87117-6008

89

This final report was prepared by the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico under Job Order 2304Y101. Lt Col Carl E. Oliver (CA) was the Laboratory Project Officer-in-Charge.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been authored by employees of the United States Government. Accordingly, the United States Government retains a nonexclusive, royalty-free license to publish or otherwise reproduce the material contained herein, or allow others to do so, for United States Government purposes.

This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify AFWL/SCP, Kirtland AFB, NM 87117-6008 to help us maintain a current mailing list.

This technical report has been reviewed and is approved for publication.

Edward A Carmona

EDWARD A CARMONA
Captain, USAF
Computer Research Scientist

FOR THE COMMANDER

Nazareno L. Rapagnani

NAZARENO L. RAPAGNANI, PhD
Chief, Communications-Computer
Systems Technology Office

DO NOT RETURN COPIES OF THIS REPORT UNLESS CONTRACTUAL OBLIGATIONS OR NOTICE ON A SPECIFIC DOCUMENT REQUIRES THAT IT BE RETURNED.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFWL-TR-89-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Air Force Weapons Laboratory 6b. OFFICE SYMBOL (If applicable) SCP	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Kirtland Air Force Base, NM 87117-6008	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2304; TASK NO Y1; WORK UNIT ACCESSION NO 01

11. TITLE (Include Security Classification)
A MODEL OF PARALLEL PERFORMANCE

12. PERSONAL AUTHOR(S)
Carmona, Edward A., Capt, USAF; and Rice, Michael D.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Jul 88 TO Dec 88	1989 April	50

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Parallel processing; Parallel computing; Parallel performance; Amdahl's Law; Parallel performance models
12	05		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report introduces a general model of parallel performance. With the goal of developing conceptual and empirical methods for characterizing and understanding parallel algorithms, new definitions of speedup and efficiency have been formulated. These definitions take into account the effects that problem size and the number of processors have on efficiency and speedup, and provide a natural and quantifiable measure of parallel performance. The terms introduced in the definitions provide new and improved interpretations of the "serial" and "parallel" fraction parameters commonly used in the literature (i.e., Amdahl's Law) to explain the behavior of parallel algorithms. The model provides a more complete characterization of parallel algorithm behavior and is used to correct apparent deficiencies in the formulation of speedup as expressed by Amdahl's Law.

Chapter 2 of this report reviews the basic definitions of speedup and efficiency and introduces new definitions of these quantities in terms of work units and presents two (over)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
EDWARD A CARMONA, Capt, USAF	(505) 844-4810	SCP

19. ABSTRACT (Continued)

illustrative examples. Chapter 3 discusses two models of parallel performance which appear in the parallel computing literature. The two formulations of speedup and efficiency are based on the notion of "serial" and "parallel" fractions and present an apparent dichotomy which must be resolved. Then, the "serial" and "parallel" fractions are expressed in terms of the new work quantities introduced in Chapter 2, providing a natural interpretation of the serial and parallel fractions of a task. The resulting equations are used to reconcile the (apparent) dichotomy between Amdahl's Law and the more recent formulation of speedup. Derivation of several fundamental laws involving the relationships between the serial and parallel fractions of a task are given and numerical evidence regarding the behavior of the serial and parallel fractions as the problem and ensemble size vary are presented. These observations are incorporated into an idealized and a general model of parallel performance.

DTIC
COPY
INSPECTED
4

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Summary

This report introduces a general model of parallel performance. With the goal of developing conceptual and empirical methods for characterizing and understanding parallel algorithms, new definitions of speedup and efficiency have been formulated. These definitions take into account the effects that problem size and the number of processors have on efficiency and speedup, and provide a natural and quantifiable measure of parallel performance. The terms introduced in the definitions provide new and improved interpretations of the "serial" and "parallel" fraction parameters commonly used in the literature (*i.e.*, Amdahl's Law) to explain the behavior of parallel algorithms. The model provides a more complete characterization of parallel algorithm behavior and is used to correct apparent deficiencies in the formulation of speedup as expressed by Amdahl's law.

Preface

We would like to thank Dr. D. Davenport from Sandia National Laboratories, Albuquerque, NM, for useful discussions and his contribution in reconfirming our interpretation of the fundamental parameters. We would also like to thank Dr. N. L. Rapagnani and Lt Col C. E. Oliver from the Air Force Weapons Laboratory, Kirtland AFB, NM, for their support in this effort. This work was supported by the Air Force Office of Scientific Research grant 2304Y101.

Contents

Section			Page
Summary			iii
Preface			iv
Contents			v
Figures			vii
Tables			viii
1.	Introduction		1
2.	Background		3
	2.1	Measures of Performance	3
	2.2	Alternate Definition of Speedup and Efficiency	4
		2.2.1 Case Study 1	6
		2.2.2 Case Study 2	7
3.	Models of Parallel Performance		14
	3.1	Two Views of Speedup	14
	3.2	Basic Laws	18
	3.3	Dynamics of s, p, s' and p'	20
	3.4	An Idealized Model of Parallel Performance	22
	3.5	General Model of Parallel Performance	25

Contents (Continued)

Section		Page
4.	Conclusions	32
	4.1 Future Research	33
References		34
Appendix A.	Dynamics of Fundamental Parameters	35

Figures

Figure		Page
1.	Task dependency and scheduling charts for Example 1.	6
2.	Cholesky data dependency chart.	8
3.	Scheduling chart for Cholesky factorization of 7X7 matrix.	10
4.	Graphs of simulated wasted work and simulated/estimated speedups.	12
5.	Constant speedup and efficiency curves.	13
6.	Speedup given by Amdahl's Law and by Sandia's reformulation.	15
7.	Box diagram of Amdahl's serial and parallel fractions.	17
8.	Realistic distribution of serial and parallel fractions.	18
9.	Serial Fraction, s, as a Function of n_p.	22
10.	Relationship between s and s'; $n_p = 10$.	27
11.	Boxed representations of parallel tasks.	29
12.	Level curves for S, E, s, and s'.	30

Tables

Table		Page
1.	Speedup and efficiency for Example 1.	7
2.	Speedup and efficiency for Example 2.	9
3.	Dynamics of fundamental parameters – Example 1.	21
4.	Dynamics of fundamental parameters – Example 2.	21
A.1.	Speedup, s, s' — Example 1.	36
A.2.	Speedup, s, s' — Example 2.	37
A.3.	Speedup, s, s' — Example 3.	38
A.4.	Speedup, s, s' — Example 4.	39
A.5.	Speedup, s, s' — Example 5.	40

1. Introduction

The field of parallel computing will become an established discipline when cost-effective high-performance parallel algorithms can be routinely implemented based on natural parallel programming models. An important issue to be resolved in the field is how to effectively analyze the performance of parallel algorithms. With the goal of developing analytical methods for characterizing and understanding parallel algorithm implementations, this report proposes new definitions of speedup and efficiency. These definitions provide a more natural, quantifiable, and multifaceted measure of the performance of parallel algorithms than current models such as Amdahl's law (Ref. 1) offer. In addition, the new parameters introduced in our definitions of speedup and efficiency provide new and improved interpretations of the "serial" and "parallel" fraction quantities frequently used in the literature to explain the behavior of parallel algorithms. These definitions take into account the effects that both the problem size and the number of processors have on efficiency and speedup and allow us to formulate and prove a number of basic laws which form the basis of this model. The model provides the foundation for future theoretical and empirical studies which will contribute to a deeper understanding of parallel algorithms.

Chapter 2 of this report reviews the basic definitions of speedup and efficiency and introduces new definitions of these quantities in terms of work units and presents two illustrative examples. Chapter 3 discusses two models of parallel performance which appear in the parallel computing literature. The two formulations of speedup and efficiency are based on the notion of "serial" and "parallel" fractions and present an apparent dichotomy which must be resolved. Then, the "serial" and "parallel" fractions are expressed in terms of the new work quantities introduced in Chapter 2, providing a natural interpretation of the serial and parallel fractions of a task. The resulting equations are used to reconcile the (apparent) dichotomy between Amdahl's law and the more recent formulation of speedup proposed in Ref. 2. Derivation of several fundamental laws involving the relationships between the serial

and parallel fractions of a task are given and numerical evidence regarding the behavior of the serial and parallel fractions as the problem and ensemble size vary are presented. These observations are incorporated into an idealized and a general model of parallel performance. Finally, Chapter 4 discusses conclusions and directions for future work.

2. Background

2.1 Measures of Performance

Speedup has been advocated as the primary measure of parallel algorithm performance. Intuitively, the speedup, S, is defined as the relative increase in speed over the serial computation as processing elements are added to the parallel computation of a given work load. Formally, this can be stated as

$$S(n_p) = \frac{T(1)}{T(n_p)} \tag{1}$$

where $T(n_p)$ is the time expended performing a task using n_p processors. Since parallel implementations may introduce computations which are unnecessary with respect to serial implementations, $T(1)$ is the time required to execute the task on a single processor using the "best" serial implementation. Clearly, the time required to execute a task depends on the number of operations that need to be performed, which, in turn, is a function of the problem size. Therefore, a more complete formulation of speedup needs to take into account the size of the problem. An alternate formulation of speedup is

$$S(n, n_p) = \frac{T(n, 1)}{T(n, n_p)} \tag{2}$$

where n denotes the problem size (this view is also formulated in Ref. 3). Ideally, one expects S to equal n_p; that is, as processing elements are added, speedup should increase at the same rate. However, this is seldom the case because of overhead introduced by the parallel implementation (Ref. 4). Therefore, efficiency is often used to measure the optimality of the speedup. Efficiency is expressed as

$$E = \frac{S}{n_p} \tag{3}$$

Because efficiency normalizes on the number of processors used, it is a more intrinsic measure of an implementation's parallel performance than speedup. A more important statistic than speedup, perhaps, is the cost of attaining a given speedup which is measured by the efficiency of the implementation. The continued reliance on speedup as the primary measure of parallel performance may be attributed to the unit of measurement that has been adopted: execution time. Using time as a measure of work has several drawbacks. First, it varies with the computer used. Second, it is simply a statistic which does not provide any particular insight about the algorithm. We need to describe the efficiency of a parallel implementation in terms of a measure of work that is quantifiable and useful for interpretations of the observable behavior of the implementation.

2.2 Alternate Definition of Speedup and Efficiency

An alternative measure of performance is provided by computational counts or unit counts based on the size of an indivisible task. This same measure of work has been used to compare serial implementations of an algorithm with order, growth rate, and complexity analysis functions such as Θ, Ω, and Big O. In these terms, the efficiency of a parallel algorithm can be defined as

$$E = \frac{wa}{we} \tag{4}$$

where wa is the work accomplished, we is the work expended, and $ww = we - wa$ is the work wasted. For a given implementation, the work wasted accounts for the time expended in the following activities:

- Waiting for other tasks to complete work
- Communication delays and/or memory contentions associated with a particular computer architecture and an implementation's communication load
- Operation redundancies introduced by the particular parallel implementation, including task activation/termination overhead

Work accomplished, wa, is a function of the problem size but is independent of the number of processors used since it refers to the amount of work that a single processor would accomplish. In terms of units of measurement, wa equals the number of operations performed by the "best" serial implementation of the algorithm. On the other hand, work wasted is a function of program size and the number of processors used since, for example, problem size affects the number of redundant operations introduced, and ensemble size affects the waiting behavior of the implementation. Therefore, Eq. 4 can be restated as

$$E = \frac{wa(n)}{ww(n, n_p) + wa(n)} \tag{5}$$

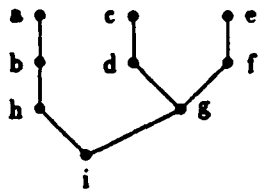
and speedup can be defined in terms of efficiency as

$$S = E * n_p = \frac{wa(n)}{ww(n, n_p) + wa(n)} * n_p \tag{6}$$

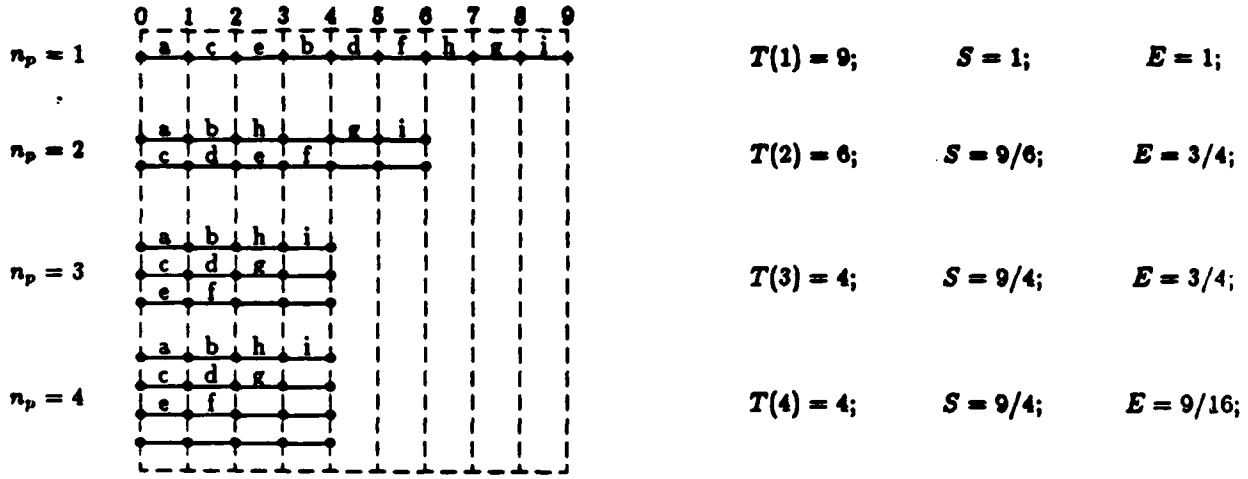
A similar formulation is introduced in Ref. 4 where efficiency is defined as a function of f_c, the ratio between communication and computation loads. In fact, Eq. 5 can be rewritten in the form:

$$E = \frac{1}{(ww/wa) + 1} \tag{7}$$

which looks very much like the formulation in Ref. 4 with $f_c = ww/wa$. The difference, however, is of practical concern, since the formulation in Ref. 4 only considers time wasted due to message delays, while the new definitions take into account the effects of load balance, task overhead, and operation redundancies. These formulations of efficiency and speedup parameterize algorithms and clearly demonstrate the effect of the individual work parameters on the overall value of the quantities. As such, the formulations may be better suited for comparative studies of parallel implementations.



(a) Task dependency graph.



(b) Scheduling chart.

Figure 1. Task dependency and scheduling charts for Example 1.

To show the applicability of these new definitions, two examples are presented. The first example is a simple artificial problem used to introduce some general concepts. The second example is based on a parallel implementation of a Cholesky matrix factorization algorithm.

2.2.1 Case Study 1

Figure 1a shows a directed acyclic graph which represents a possible computational task. Nodes in the graph represent indivisible tasks, and the arcs represent task dependencies. For example, task *b* cannot start until task *a* is accomplished. Figure 1b shows the scheduling of independent tasks on 1, 2, 3, and 4 processors. For simplicity, each task is assumed to accomplish the same amount of work and no time is wasted due to communication delays or operation redundancies.

The multitasked implementations indicate portions of wasted effort, ww, due to task dependencies as unlabeled segments. The speedup and efficiency of each implementation (shown

Table 1. Speedup and efficiency for Example 1.

n_p	ww	wa	we	$E = wa/we$	$S = En_p$
1	0	9	9	1	1
2	3	9	12	3/4	3/2
3	3	9	12	3/4	9/4
4	7	9	16	9/16	9/4
5	11	9	20	9/20	9/4

on the right of each graph) are computed using the conventional Eqs. 1 and 3. Table 1 shows speedup and efficiency computed using the new definitions as expressed by Eqs. 5 and 6. As can be seen, the speedup and efficiency computed by the different definitions coincide.

Note that a maximum speedup of 9/4 is reached with three processors and that adding processors only increases the work wasted which simply decreases the efficiency. Also, note that the maximum speedup of 9/4 equates to the reciprocal of the ratio of the length of the longest dependency chain in the dependency graph (4) to the total number of tasks (9). Chapter 3 shows that this ratio represents the speedup bound predicted by Amdahl's law.

2.2.2 Case Study 2

A common algorithm applied to structural analysis, hydrodynamics, and least square problems is the Cholesky factorization of an n_{th} order square matrix. The general solution of the Cholesky factorization is a specialized form of the more common Gaussian elimination algorithm (see Ref. 5). The Cholesky method takes advantage of symmetry properties of the matrix to reduce the complexity of factorization from an $O(4n^3/3)$ problem as solved by the Gaussian elimination method (Ref. 6) to an $O(n^3/3)$ problem (Ref. 7). Depending on how the looping variables are arranged, there are six different implementations of this algorithm (Ref. 8). However, the following discussion concentrates on the ijk form of Cholesky.

The algorithm is designed for a shared memory architecture and relies on a dynamic scheduling scheme to assign rows of the matrix to available processors. That is, when a processor

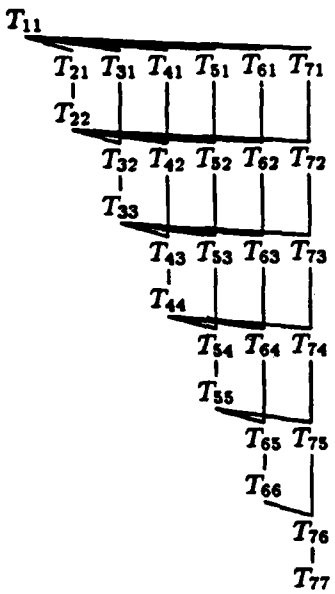


Figure 2. Cholesky data dependency chart.

becomes available, the next row to be computed is assigned to that processor. The order in which rows are assigned to processors is contingent on (1) the data dependencies associated with the algorithm, and (2) the manner in which the algorithm is implemented.

The data dependencies for a 7×7 matrix are shown in Fig. 2. Each node T_{ij} represents the ij_{th} element of the resultant matrix. These data dependencies are inherent in the algorithm. The synchronization scheme adopted by a given implementation must ensure that these inherent dependencies are preserved.

The ijk form of Cholesky solves the system of equations row–wise. Observe that, in order to factor the i_{th} row, the previous $i-1$ rows are required. More specifically, to compute the j_{th} element of any row, the elements of the j_{th} row are required. Thus, one possible implementation can have task i wait for the j_{th} row to be computed before the ij_{th} element is computed. The basic algorithm is similar to the implementation of the column-Cholesky in Ref. 9 and is as follows:

```
for task i
  for j := 1 to i-1 do
    wait for jth row
```

Table 2. Speedup and efficiency for Example 2.

n_p	ww	wa	we	$E = wa/we$	$S = En_p$
1	0	168	168	1	1
2	44	168	212	84/106	84/53
3	126	168	294	84/147	84/49
4	224	168	392	84/196	84/49
5	322	168	490	84/245	84/49

```
    for k := 1 to j-1 do
       a(i,j) := a(i,j) - a(i,k)*a(j,k);
    end
    a(i,j) = a(i,j)/a(j,j);
    a(i,i) = a(i,i) - a(i,j)*a(i,j);
  end
    a(i,i) = sqrt(a(i,i));
end
```

In order to create the scheduling chart for the Cholesky algorithm, assume that each additive and multiplicative operation corresponds to one unit of work, while divide and square root operations cost two work units each. Based on these assumptions, it takes $2j$ work units to compute the j_{th} element of any row. Scheduling charts for implementations on $1 - 4$ processors are shown in Fig. 3 with their respective speedup and efficiency (shown on the right of each graph) computed using Eqs. 1 and 3. The scheduling charts do not take into account communication delays or time required for task switching. Table 2 shows speedup and efficiency computed with the alternate definitions. Once again, speedup and efficiency statistics computed by the new definitions and the standard definitions coincide.

One can generalize the effects n and n_p have on speedup and efficiency for the above implementation by deriving closed-form equations which estimate wa and ww. A computational count shows that there are $n^3/3 - n/3$ multiplications and additions and $(n^2 + n)/2$ divisions and square roots. Thus the work accomplished is $wa = (n^3/3 + n^2 + 2n/3)$ computational

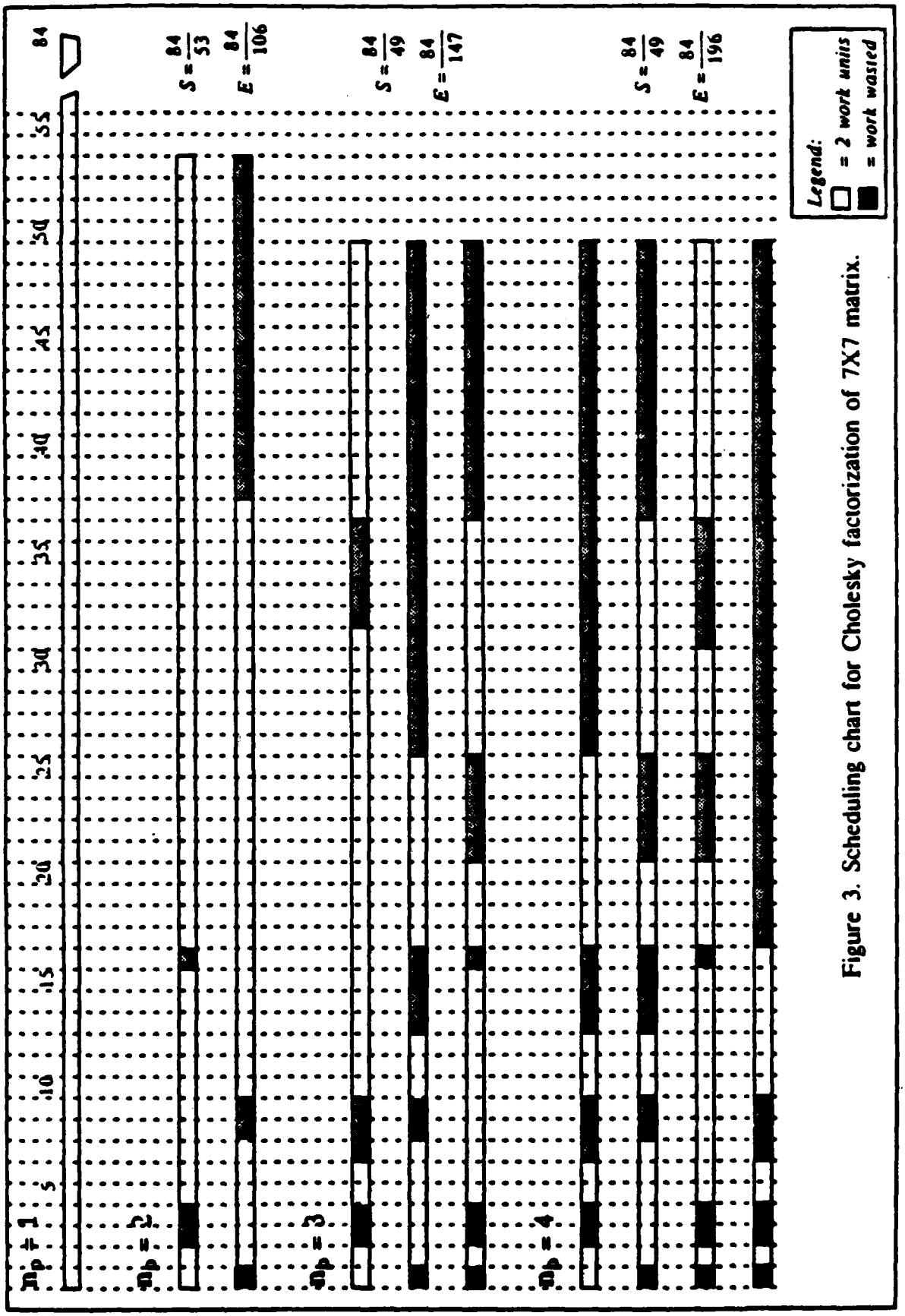


Figure 3. Scheduling chart for Cholesky factorization of 7X7 matrix.

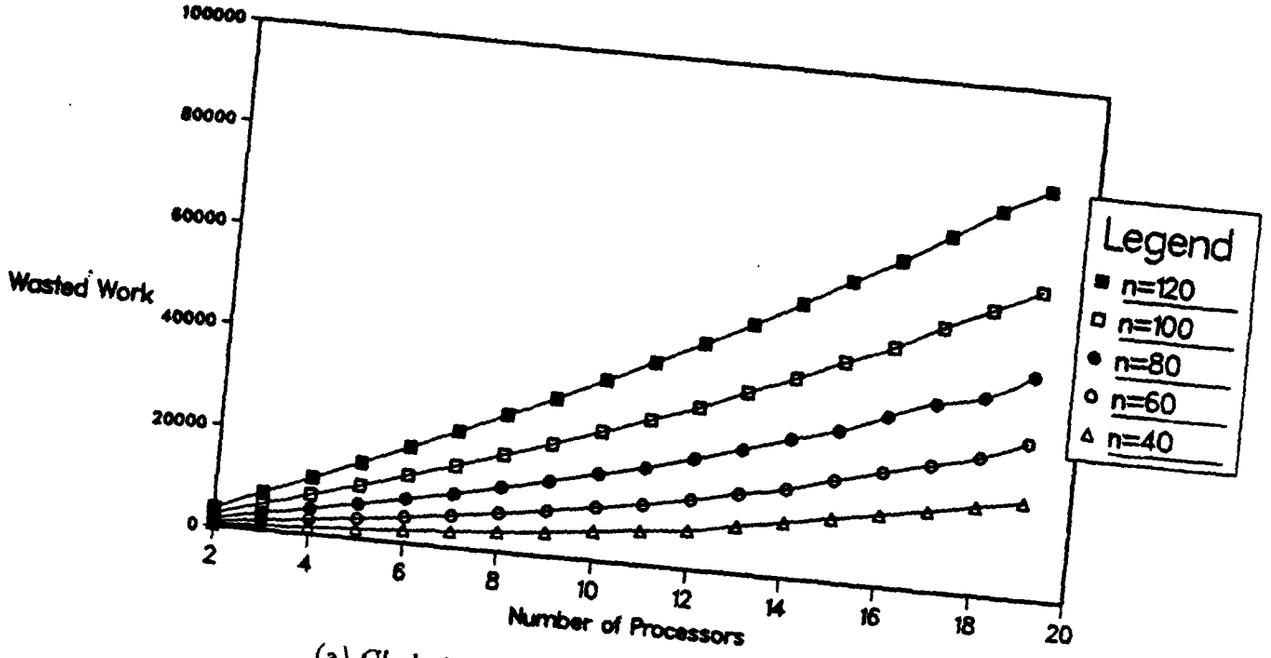
units (division and square root operations count for two operations each). This expression is an exact measure of work accomplished based on our assumptions.

To derive a closed–form equation for the work wasted, however, is not so straightfoward. The effect of problem and ensemble size on the waiting behavior of an implemention can be somewhat difficult to measure. To estimate wasted work, a simulation based on the same assumptions made above concerning the cost of each operation was performed. Based on the simulation, a statistical analysis showed that for $n > 50$, ww can be approximated by the expression

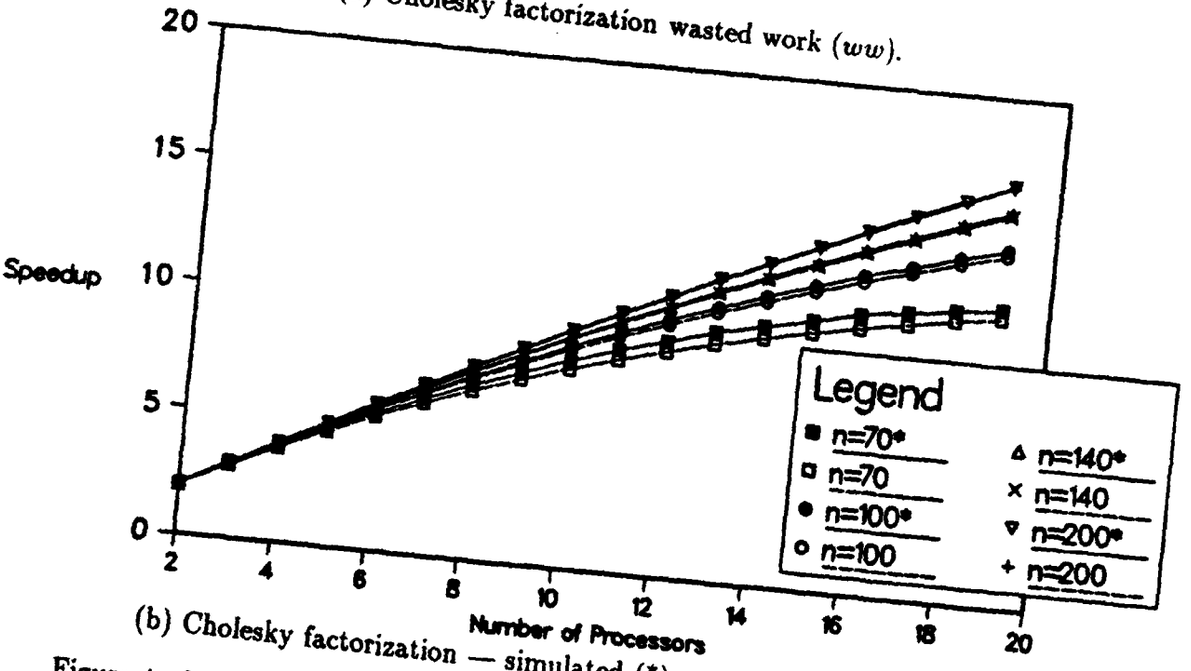
$$ww \approx \left[(38 + n/10)n_p + \frac{n^2}{4} - 149\right](n_p - 1)$$

Figure 4a shows a graph of simulated work wasted, while Fig. 4b shows simulated speedup and estimated speedup using the approximations of wa and ww. Based on these approximations, several observations can be made. First, as the size of the problem increases, speedup increases for a given ensemble size. This is due to an increase in efficiency as problem size increases. Second, as the number of processors increases, efficiency decreases for a given problem size. Figures 5a and 5b are graphs of constant speedup and efficiency curves based on our estimates of ww and wa. These curves show that a given speedup can be attained from different combinations of problem and ensemble size.

As can be seen, the waiting behavior of an implementation can be determined analytically by using closed-form expressions of the new fundamental parameters. However, in what follows, the relationship between the new definitions and other more known definitions of speedup and efficiency are explored.

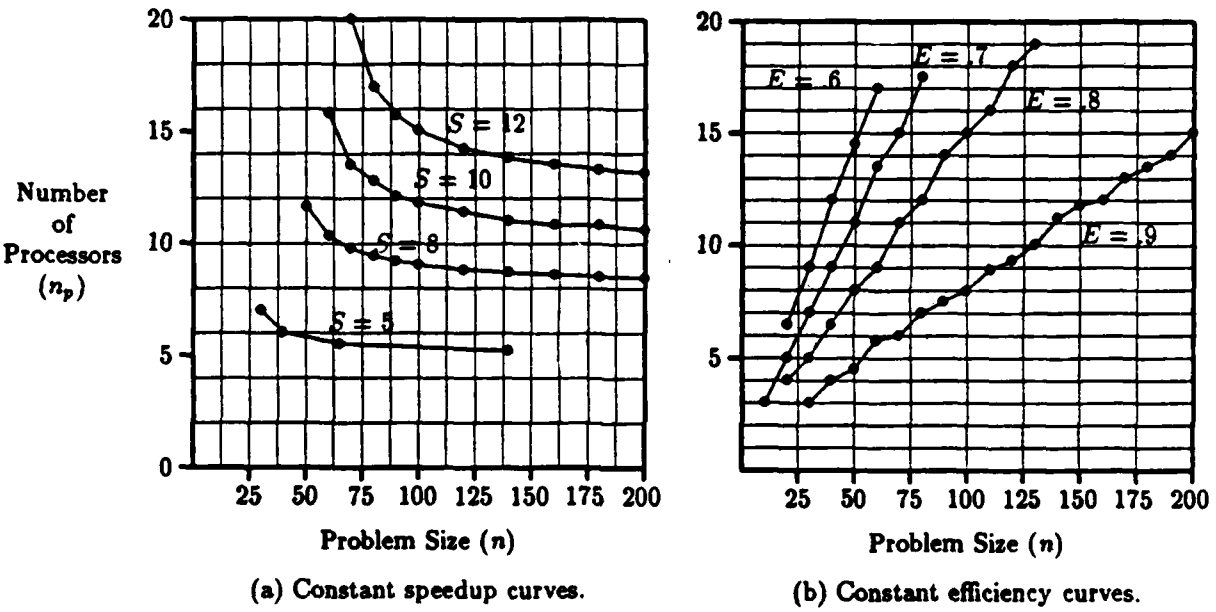


(a) Cholesky factorization wasted work (ww).



(b) Cholesky factorization — simulated (*) vs. estimated speedup.

Figure 4. Graphs of simulated wasted work and simulated/estimated speedups.



(a) Constant speedup curves.

(b) Constant efficiency curves.

Figure 5. Constant speedup and efficiency curves.

3. Models of Parallel Performance

3.1 Two Views of Speedup

The parallel computing literature contains a number of different formulations of speedup and efficiency based on the "serial" and "parallel" fractions of a given task. This section examines two different points of view for modeling performance of a parallel system. From the first perspective, the speedup is viewed in the context of the execution time of the serial task on multiple processors. From the second perspective, the speedup is viewed in the context of the execution time of the parallel task on a single processor.

The first point of view is expressed by Amdahl's law (Ref. 1) which states that if s is the fraction of time spent on the serial portion of a task and p is the fraction of time spent on portions of the task that can be executed in parallel (*i.e.*, $s + p = 1$), then the time required for a parallel system to execute the task is $(s + p/n_p) * T(1)$. Based on these definitions of s and p, and using Eq. 1, speedup can be expressed as

$$S_{amdahl} = \frac{(s+p) * T(1)}{(s + p/n_p) * T(1)} = \frac{s+p}{s+p/n_p} = \frac{1}{s+p/n_p} \tag{8}$$

where speedup has been normalized with respect to time. Figure 6a shows the effect s has on speedup for a given number of processors. The Amdahl point of view is that the serial portion of an algorithm bounds the maximum speedup that can be attained as processing elements are added to the computation. Therefore, as $n_p \rightarrow \infty$, the term $p/n_p \rightarrow 0$, so speedup is asymptotically bounded by $1/s$. This formulation also implies that for a given problem size, the maximum speedup attainable is not reached until an infinite number of processors are used.

The second point of view was recently formulated by researchers at Sandia National Laboratories. In Ref. 2, Amdahl's law is restated as follows: let $\hat{s}$ be the portion of time spent

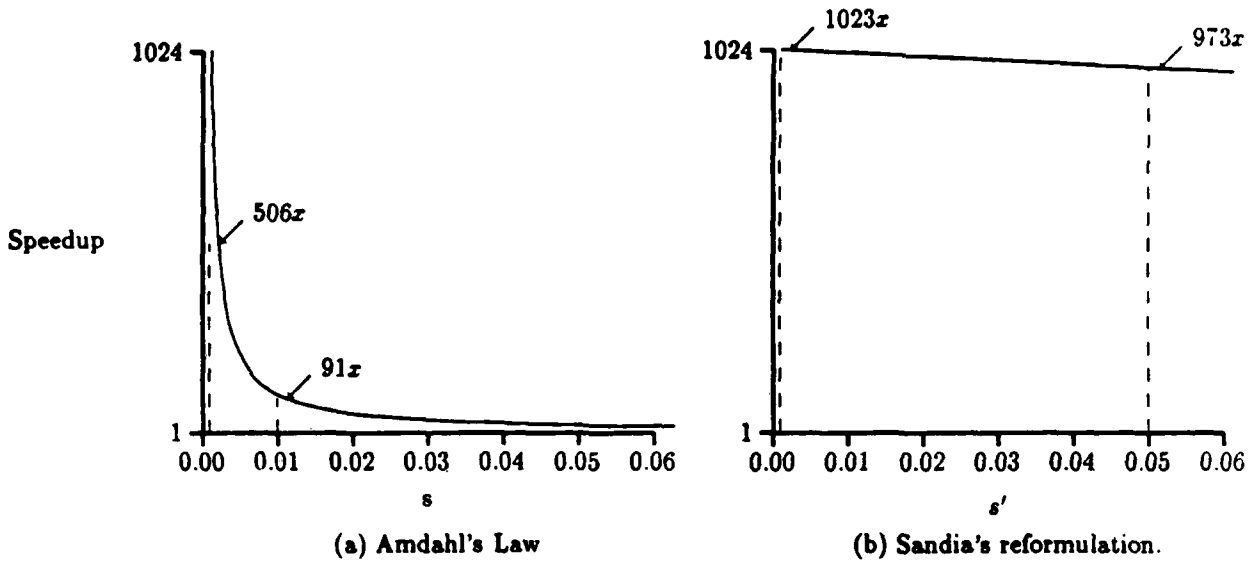


Figure 6. Speedup given by Amdahl's Law and by Sandia's reformulation.

performing serial work on the parallel system executing the task, and let p' be the portion of time spent performing parallel work on the parallel system (*i.e.*, $s' + p' = 1$). Then the time required by a single processor to perform the task is $(s' + p'n_p) * T(n_p)$. Therefore, speedup can be expressed as

$$S_{sandia} = \frac{(s' + p'n_p) * T(n_p)}{(s' + p') * T(n_p)} = \frac{s' + p'n_p}{s' + p'} = s' + p'n_p \tag{9}$$

Figure 6b shows a sample graph of speedup as defined by Eq. 9 for $n_p = 1024$. The apparent dichotomy between the two definitions of speedup is that S_{amdahl} predicts modest upper bounds on speedups for serial fractions in the range $0.01 - 0.04$, while S_{sandia} predicts potential speedups of 10 to 40 times greater magnitude in the same range. Because both laws are based on a common definition of speedup (Eq. 1), it is striking that the two formulations are so distinct. However, if it is assumed that speedup definitions (Eqs. 6, 8, and 9) are equivalent, the apparent dichotomy between the two points of view can be reconciled by expressing the serial and parallel fractions in terms of the parameters ww, wa, we, and the number of processors, n_p. In particular, it follows that the two different formulations of speedup can be considered equivalent, assuming the fractional entities are expressed in terms of the work parameters and n_p. Equating Eqs. 6 and 8, one obtains:

$$
\begin{aligned}
\frac{1}{s+p/n_p} &= \frac{wa}{wa+ww} * n_p \\
\frac{wa+ww}{wa} &= n_p\left(s+\frac{1-s}{n_p}\right) \\
1+ww/wa &= s(n_p)+1-s \\
ww/wa &= s(n_p-1) \\
s &= \frac{ww}{wa}\left(\frac{1}{n_p-1}\right) \qquad (n_p \geq 2)
\end{aligned} \tag{10}
$$

Therefore, s can be interpreted as the *distribution across the additional processors of the ratio of work wasted to work accomplished.* Equating Eqs. 6 and 9, one obtains

$$
\begin{aligned}
n_p * \frac{wa}{wa+ww} &= s'+(1-s')n_p \\
&= s'(1-n_p)+n_p \\
n_p * \frac{wa}{wa+ww} - n_p &= s'(1-n_p) \\
n_p\left(\frac{wa}{wa+ww}-1\right) &= s'(1-n_p) \\
n_p\left(-\frac{ww}{we}\right) &= s'(1-n_p) \\
s' &= \frac{ww}{we} * \frac{n_p}{n_p-1} \qquad (n_p \geq 2)
\end{aligned} \tag{11}
$$

Therefore, s' can be interpreted as a *collective wasted effort, $n_p * \bar{s}$, where $\bar{s}$ is the distribution across the additional processors of the ratio of work wasted to work expended.* Notice that both interpretations of s and s' are undefined for $n_p = 1$. This reflects the fact that it is inconsistent to consider serial and parallel portions of a task in a strictly serial context; the fractions have meaning only in a parallel context. In addition, since ww and wa are functions of n and n_p, it follows that the parameters s, s', p, and p' are not only functions of the problem size, but also functions of the number of processors used. Although this assertion seems intuitively inconsistent with the *a priori* definitions of these parameters, it is based on both theoretical considerations and empirical evidence.

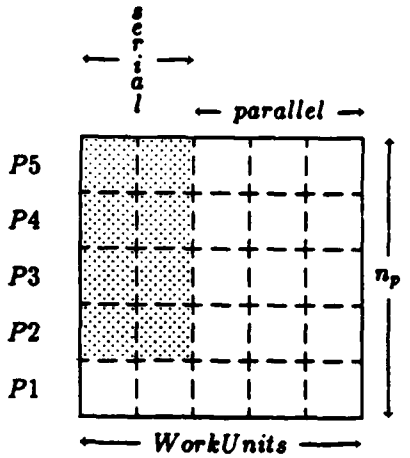
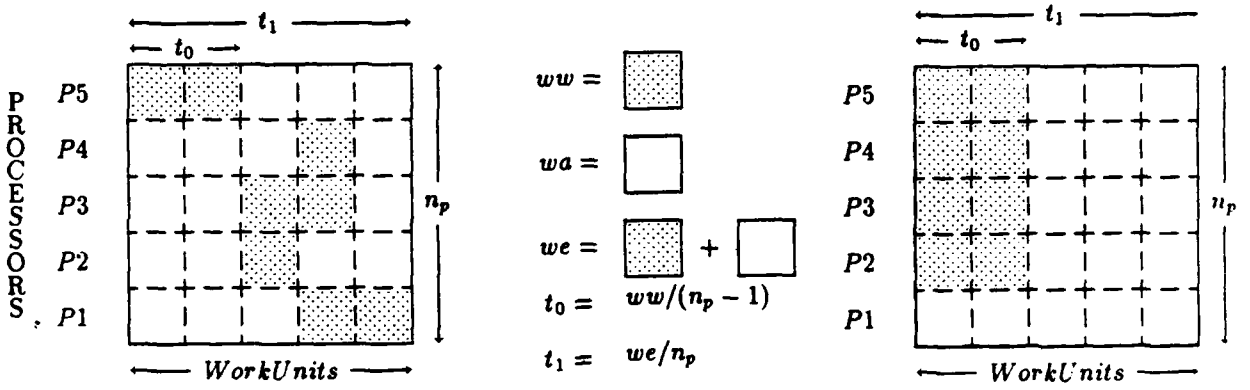


Figure 7. Box diagram of Amdahl's serial and parallel fractions.

A useful diagram for understanding the meaning of s, s', p, p' can be developed as follows: based on Amdahl's definitions of s, there exists a portion of an algorithm that must be executed serially and another portion that can be executed in parallel. For five processors the idealized work distribution may have the form shown in Fig. 7 where the shaded region indicates wasted work. A more realistic distribution of work for a multiprocessor system is depicted in Fig. 8a where, strictly speaking, there is no exclusively serial portion of work. Assume, for purposes of illustration, that in a general parallel system with n_p processors, the wasted work throughout the total execution time is initially grouped on the last $n_p - 1$ processors during the time period $0 \leq t \leq t_0 = ww/(n_p - 1)$. Furthermore, suppose that the total work, we, is expended over the time period $0 \leq t \leq t_1 = we/n_p$. We can now interpret the parameters s, s', p, p' in terms of relative areas of the new diagram, Fig. 8b. For example, s is the ratio of the width of t_0 to the length of the total white area, wa, while s' is the ratio of the width of t_0 to the total width of the box, t_1. Similar interpretations can be given for p and p'. In addition, since the diagram is expressed in terms of ww, wa, and n_p, one can derive Eqs. 10 and 11 from geometric considerations. For example, since $s = t_0/wa$ and $t_0 = ww/(n_p - 1)$ then $s = [ww/(n_p - 1)]/wa$ which is the same interpretation of s given by Eq. 10. The parameters s', p, and p' can be derived similarly. These graphical representations of tasks provide insight about the properties of s, s', p, and p'. In this parallel framework these parameters have real meaning. Amdahl's s parameter has an averaging effect in that



(a) Distribution of wasted work.

(b) Wasted work redistributed.

Figure 8. Realistic distribution of serial and parallel fractions.

it represents the portion of work which would have to be done 100 percent serially so that a portion p of the work could theoretically be performed with all processors working at 100 percent efficiency. Generally, as the number of processors is increased, work wasted changes and the fractions s and p change.

3.2 Basic Laws

Using the expressions for s and s' found in Section 3.1, several fundamental laws can be derived which illustrate the relationships between the respective serial and parallel fractions. For convenience, let $a = wa/we = E$ and let $w = ww/we$. Then Eqs. 10 and 11 can be written in the following form:

$$s = (w/a)(1/n_p - 1) \quad (n_p \geq 2) \tag{12}$$

$$s' = w[n_p/(n_p - 1)] \quad (n_p \geq 2) \tag{13}$$

Using these equations, the following fundamental laws can be derived:

$$\boxed{\text{Speedup} = \begin{cases} s'/s & \text{if } ww > 0 \\ n_p & \text{if } ww = 0 \end{cases} \quad \text{Efficiency} = \begin{cases} p'/p & \text{if } ww > 0 \\ 1 & \text{if } ww = 0 \end{cases}} \tag{14}$$

Proof: Using Eqs. 12 and 13, one obtains

$$\frac{s'}{s} = \frac{wn_p}{n_p - 1} * \frac{a(n_p - 1)}{w} = an_p = En_p = S$$

Using the definitions of p and p', Eq. 11, and the fact that $E = 1 - w$, one derives

$$\begin{aligned} p' &= 1 - s' \\ p' &= 1 - w\left(\frac{n_p}{n_p - 1}\right) \\ &= 1 - w\left(1 + \frac{1}{n_p - 1}\right) \\ &= (1 - w) - a\left[\frac{w}{a}\left(\frac{1}{n_p - 1)}\right)\right] \\ &= E - Es \\ &= E(1 - s) \\ p' &= Ep \end{aligned}$$

The basic law says that the serial fractions of a task, which have been used to define speedup, are related precisely by that speedup! The law also states that the parallel fractions of a task which have been used in defining speedup are related precisely by the efficiency of the parallel system*. Also, since $E \leq 1$ and $S \leq n_p$, the above proof establishes the following inequalities:

$$s' \leq sn_p \tag{15}$$

$$p' \leq p \tag{16}$$

*The basic laws can also be derived by expressing s and s' in terms of S in Eqs. 8 and 9 and taking the corresponding ratios.

The preceding work leads to a more unified view of speedup and important reinterpretations of the parameters affecting the parallel performance of algorithms. The unified view of speedup is provided as follows: if the expressions for s and s' are substituted in Eqs. 8 and 9, respectively, the formulations of speedup in Ref. 1 and Ref. 2 coincide with the new definition of speedup. In addition, using the basic law, the two speedup graphs shown in Fig. 6 can be correlated. For purposes of illustration, assume speedup is a linear function of $s' : S = -ms' + b$, as proposed in Ref. 2. Substituting $s' = sS$ and rearranging terms, one establishes that speedup is a hyperbolic function of $s : S = b/(1 + ms)$, as proposed in Amdahl's formulation of speedup. Therefore, the apparent dichotomy between the different formulations of speedup is resolved. In addition, the relationships between s and s' are easily derived from Eqs. 8, 9, and 14.

$$s' = \frac{s}{s + (1 - s)/n_p} \tag{17}$$

$$s = \frac{s'}{s' + (1 - s')n_p} \tag{18}$$

In order for the different formulations of speedup to remain consistent, it is necessary to treat the serial and parallel fraction parameters as dynamic entities with respect to n_p. If one insists on viewing these fractions as constants for a given problem size, the formulations of speedup found in Ref. 1 and Ref. 2 only describe the limiting behavior of a general parallel system. Therefore, from this point on, the parameters s, s', p, and p' will be treated as variables which depend on n and n_p.

3.3 Dynamics of s, p, s' and p'

This section illustrates the behavior of s and s' with respect to n, and n_p. These observations will support the assumptions made in the models of parallel performance presented in Sections

Table 3. Dynamics of fundamental parameters - Example 1.

n_p	s	s'	$S = s'/s$	p	p'	$E = p'/p$
2	0.333	0.500	1.52	0.667	0.500	0.750
3	0.167	0.375	2.25	0.833	0.625	0.750
4	0.259	0.583	2.25	0.741	0.417	0.563
5	0.306	0.688	2.25	0.694	0.313	0.450

Table 4. Dynamics of fundamental parameters - Example 2.

n_p	s	s'	$S = s'/s$	p	p'	$E = p'/p$
2	0.262	0.415	1.585	0.738	0.585	0.792
3	0.375	0.643	1.714	0.625	0.357	0.571
4	0.444	0.762	1.714	0.556	0.238	0.429
5	0.479	0.821	1.714	0.521	0.179	0.343

3.4 and 3.5. Tables 3 and 4 show sample values of the serial and parallel fractions as expressed by Eqs. 10 and 11 for the examples presented in subsections 2.2.1 and 2.2.2 and shows the computation of speedup and efficiency based on the basic law expressed in Eq. 14.

Notice that even though the maximum value of speedup is attained, the values of s and s' (and therefore p and p') continue to change as n_p increases. This occurs because the work wasted increases as n_p increases. A straightfoward analysis of example 1 shows that for $n_p \geq 3$, $ww = 4n_p - 9$ and $we = (4n_p - 9) + wa = 4n_p$. Therefore, $s = (4n_p/9 - 1)/(n_p - 1)$ and $s' = (n_p - 9/4)/(n_p - 1)$ for $n_p \geq 3$. It follows that

$$\lim_{n_p \to \infty} s = 4/9 = 1/MaxSpeedup$$

and

$$\lim_{n_p \to \infty} s' = 1$$

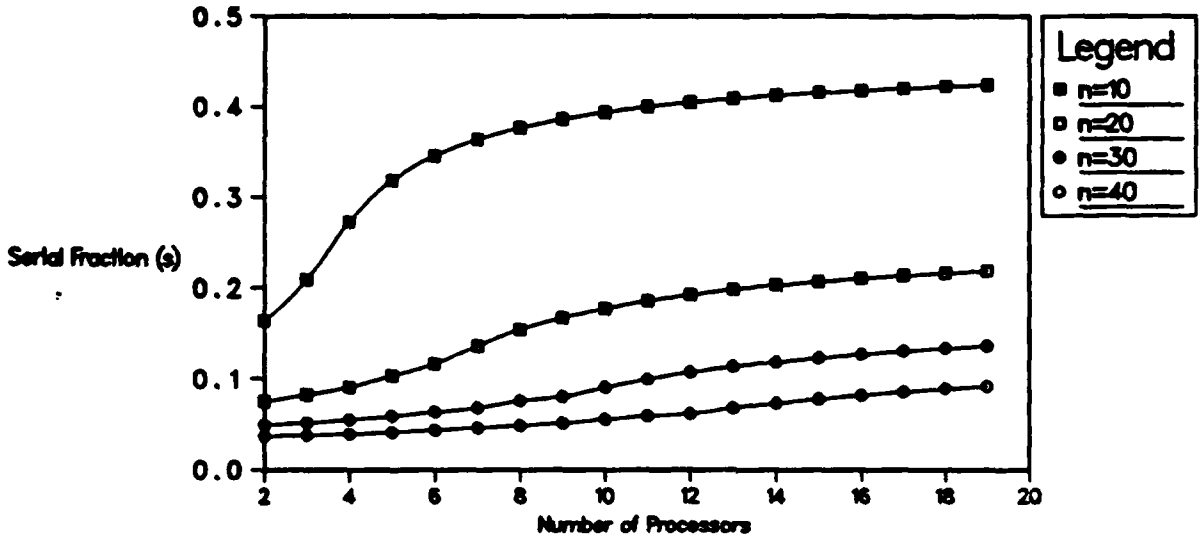


Figure 9. Serial Fraction, s, as a Function of n_p.

Based on the simulation described in subsection 2.2.2, the graphs in Fig. 9 of the $s(n, n_p)$ curves were obtained. These curves illustrate the following facts about the simulation of the Cholesky algorithm. First, as problem size increases, respective s values decrease (fixed n_p). Second, as the number of processors increases, the s values increase (fixed n). Third, the points of inflection of the curves correspond to the number of processors at which the maximum speedup is attained. This behavior is also shown by the tables found in the appendix, which were computed from the timing data found in Ref. 2 and Ref. 3 for certain parallel numerical algorithms implemented on a 1024–node hypercube.

3.4 An Idealized Model of Parallel Performance

The preceding theoretical and empirical examples suggest certain characteristic behavior of the parameters s, p, s', and p'. For example, for a fixed number of processors n_p, as the problem size increases, Tables 3, 4, and those in the appendix show that s decreases and S increases. Also, for a fixed n, as n_p increases, the tables seem to show that s' approaches 1. Finally, for a given n, the tables seem to indicate that the values of s fluctuate until a certain n_p is reached, at which point s begins to increase to an asymptotic limit. These observations,

in connection with the laws presented in Section 3.2, form the basis for the model of the performance of parallel algorithms that will be presented in the next section. However, before presenting the model, the limiting or asymptotic behavior of the fundamental parameters will be discussed.

The effect of scheduling tasks to processors can be thought of as follows: as processors are added to the computation, the original serial task is partitioned into smaller and smaller independent portions while preserving the inherently sequential portions of the algorithm. There is a point where further partitioning is not possible without conflicting with the inherent dependencies. Thus, it is useful to base the discussion of asymptotic behavior on the fundamental assumption that there is a largest portion of a task, h, which is indivisible (in the parallel sense)*. For instance, in Example 2.2.1, h is the ratio of the depth of the dependency graph (d) and the total number of atomic tasks (nodes in the graph) (A), so $h = d/A = 4/9$. A so-called "embarrasingly parallel" or purely parallel algorithm may have $d = 1$, the time required to execute a single computation or instruction. In terms of the box diagrams used in Section 3.1, the width of a box will never be less than d work units, restricting any further speedup. Pursuing this analogy, in an optimum implementation where d has been reached, the work expended is $we = dn_p$ and the work accomplished is $wa = A$. Therefore, $ww = we - wa = dn_p - A$. Using Eq. 10, the "idealized" s, denoted $\hat{s}$, is

$$\begin{aligned} \hat{s} &= \frac{dn_p - A}{A}\left(\frac{1}{n_p - 1}\right) \\ &= \frac{d/An_p - 1}{n_p - 1} \\ \hat{s} &= \frac{hn_p - 1}{n_p - 1} = h - \frac{1 - h}{n_p - 1} \end{aligned} \tag{19}$$

The preceding formulation is valid only when the number of processors used equals or exceeds the breadth of the dependency graph, or when the number of processors needed to attain

*This fraction is conceptually different from Amdahl's s in that it represents a portion of work that can be executed in parallel with other tasks. It represents work that must be performed sequentially, not serially.

the maximum speedup is reached. This reflects the fact that $\hat{s}$ models only the asymptotic behavior of s. The precise relationship between s and $\hat{s}$ is obtained by using Eq. 8 and writing s in terms of speedup: $s = [(n_p/S) - 1]/(n_p - 1)$. Then

$$\begin{aligned} s - \hat{s} &= \frac{1}{n_p - 1}\left[\frac{n_p}{S} - 1 - (hn_p - 1)\right] = \frac{n_p}{n_p - 1}\left(\frac{1}{S} - h\right) \\ s &= \hat{s} + \frac{n_p}{n_p - 1}\left(\frac{1}{S} - h\right) \end{aligned} \tag{20}$$

Once the maximum speedup is obtained for a given implementation, $S = A/d = 1/h$. It follows from Eq. 20 that $s = \hat{s}$, so

$$\lim_{n_p \to \infty} s = \lim_{n_p \to \infty} \hat{s} = h \tag{21}$$

Based on Eq. 21, the limiting behavior of s actually defines a number which can be interpreted as the largest irreducible fraction of a task that must be executed in a sequential (not serial) fashion. Furthermore, based on the assumption that $\lim_{n_p \to \infty} s = s_\infty$ exists, the following results can be established:

$$(i) \quad \lim_{n_p \to \infty} S = \frac{1}{s_\infty} \qquad (ii) \quad \lim_{n_p \to \infty} s' = 1 \tag{22}$$

To establish (i), based on Eq. 8,

$$\begin{aligned} |\, \tfrac{1}{S} - s \,| &= |\, s + (1 - s/n_p) - s \,| \\ &= (1 - s)/n_p \quad < 1/n_p \end{aligned} \tag{23}$$

Therefore, $1/S - s \to 0$ as $n_p \to \infty$. To establish (ii), by Eq. 14,

$$\begin{aligned} s' &= Ss, \qquad \text{so} \\ \lim_{n_p \to \infty} s' &= \lim_{n_p \to \infty} S * \lim_{n_p \to \infty} s = (1/s_\infty) * s_\infty = 1 \end{aligned}$$

One can also derive "idealized" versions of s', p, and p' using the preceding type of reasoning. For example, using Eqs. 9 and 11 one derives $\hat{s}' = (s_\infty n_p - 1)/s_\infty(n_p - 1)$, so that $S = \hat{s}'/\hat{s} = 1/s_\infty$ (for sufficiently large n_p). These idealized versions of the basic parameters constitute an asymptotic model of parallel performance. As indicated by Eqs. 19 and 20, the actual versions of the parameters differ from the parameters in the asymptotic model by error terms which approach zero as $n_p \rightarrow \infty$ (provided $\lim_{n_p \rightarrow \infty} s = s_\infty$ exists).

The model predicts that for a given problem size, even though s is changing with n_p, it asymptotically approaches a maximum which in turn determines the speedup that can be attained. This is in accordance with Amdahl's law. However, the model also says that unless these fundamental parameters are treated as dynamic entities, Amdahl's formulation of speedup is only correct for $n_p = \infty$. The argument can be made that, for the *a priori* definitions of the fundamental parameters (*i.e.*, s, p, etc.), the formulations of speedup expressed in Ref. 1 and Ref. 2 are correct. However, the definitions place the overly restrictive assumption that work represented by p is equally partitioned among all processors. If, however, the new interpretations presented herein are given to the fundamental parameters, they can serve as parameters in a general model of parallel performance.

3.5 General Model of Parallel Performance

This section presents a group of assumptions and their consequences about the fundamental parameters which collectively represent a dynamic model of parallel performance. There is one primary assumption, $A1$, and two secondary assumptions, a_1 and a_2.

($A1$) The definitions of speedup and efficiency presented in this paper are equivalent. That is, speedup and efficiency are absolute, not relative entities.

A partial justification for this assumption comes from the observation that each of the three definitions of speedup (Eqs. 6, 8, and 9) was motivated by the original concept of speedup as

the ratio of the execution times required to perform the task in serial and parallel modes (Eq. 1). The reconciliation of the different views of speedup based on $(A1)$ led to the following consequences (Eqs. 10, 11, and 14):

$(C1)$ $s = (ww/wa)\left(\frac{1}{n_p - 1}\right)$ $\qquad s' = (ww/we)\left(\frac{n_p}{n_p - 1}\right)$

$(C2)$ Speedup $= \begin{cases} s'/s & \text{if } ww > 0 \\ n_p & \text{if } ww = 0 \end{cases}$ $\qquad$ Efficiency $= \begin{cases} p'/p & \text{if } ww > 0 \\ 1 & \text{if } ww = 0 \end{cases}$

The empirical and theoretical evidence presented in Section 3.3 suggests possible behavior for the parameters s and s' that will be incorporated into the model as the following secondary assumptions:

(a_1) s is an increasing function of n_p (for constant problem size n and n_p sufficiently large).

(a_2) s is a decreasing function of n (for a constant number of processors n_p and sufficiently large n).

Using reasoning similar to the type presented in Section 3.1, one can show that $(A1)$ and (a_1) imply the following consequences:

$(C3)$ $\qquad \lim_{n_p \to \infty} s = s_\infty$ exists and $\lim_{n_p \to \infty} S = 1/s_\infty$.

$(C4)$ s' is an increasing function of n_p and $\lim_{n_p \to \infty} s' = 1$.

$(C5)$ E is a decreasing function of n_p and $\lim_{n_p \to \infty} E = 0$.

Since s is an increasing function of n_p and $s \leq 1$, $(C3)$ follows. To establish $(C4)$, by Eq. 17 one can write

$$s' = \frac{1}{1 + (1/s - 1) * (1/n_p)} = \frac{1}{1 + r}$$

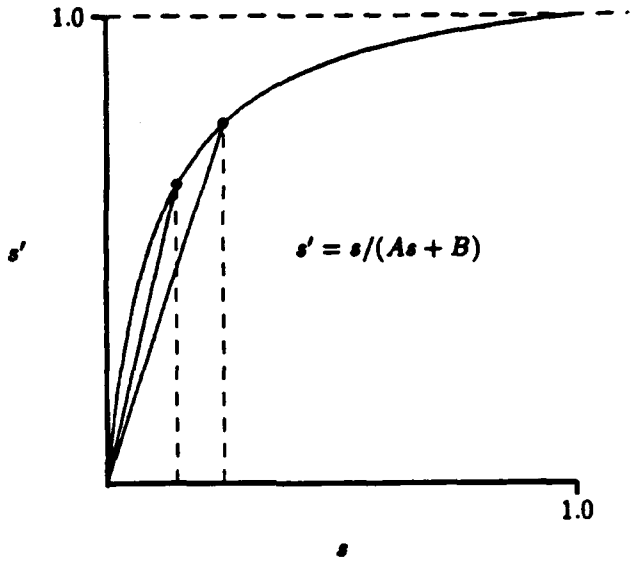


Figure 10. Relationship between s and s'; $n_p = 10$.

By (a_1), as n_p increases, s increases, so $(1/s - 1)$ decreases. Since $1/n_p$ also decreases, r decreases so s' increases. By $(C3)$, s_∞ exists and is nonzero, so $\lim_{n_p \to \infty} r = 0 \Rightarrow \lim_{n_p \to \infty} s' = 1$, which establishes $(C4)$. To establish $(C5)$, write the definition of efficiency in the form

$$E = \frac{S}{n_p} = \frac{1}{n_p[s + (1-s)/n_p]} = \frac{1}{s(n_p - 1) + 1}$$

By (a_1), as n_p increases, s increases, so $s(n_p - 1) + 1$ increases, which implies that E decreases. By $(C3)$, s_∞ exists and is nonzero, so $\lim_{n_p \to \infty} E = 0$.

Assumption (a_2) involves the parameter n which is not explicit in any of the expressions that have been presented. However, assuming a constant number of processors, one can state a basic relationship between s' and s that can be used to derive consequences of (a_2). As previously noted, one can write s' in the following form:

$$s' = \frac{s}{s(1 - 1/n_p) + 1/n_p} = \frac{s}{As + B}$$

where $A = 1 - 1/n_p$ and $B = 1/n_p$. Viewing s and s' as real variables in the range $(0, 1)$, one computes $ds'/ds = B/(As + B)^2 > 0$ and $d^2s'/ds^2 = -2AB/(As + B)^3 < 0$. Then one

obtains the graphical representation of the relationship between s and s' found in Fig. 10 where the average slope of the graph at a particular s represents speedup. Using Fig. 10 and (a_2), one can derive the following consequences (for simplicity the parenthetical remarks stated in (a_2) are omitted).

$(C6)$ s' is a decreasing function of n.

$(C7)$ S and E are increasing functions of n.

Since $ds'/ds > 0$, it follows from (a_2) that $(C6)$ holds. Furthermore, since S is the average slope of a concave downward graph ($d^2s'/ds^2 < 0$), it follows that as s decreases, S increases. Therefore, by (a_2), as n increases, S also increases. Since $E = S/n_p$ and n_p is fixed, it follows that E increases as n increases, so $(C7)$ is established.

The assumptions and consequences presented above can be interpreted in terms of graphical diagrams like the ones presented in Subsection 2.2.2. This interpretation provides an intuitive explanation of the model in terms of the behavior of the work quantities wa, ww, and we.

As usual, in Fig. 11, the shaded area represents work wasted, ww, and the white areas represent the work accomplished, wa. Diagram 1 illustrates the model for increasing n_p and diagram 2 illustrates the model for increasing n. For sample interpretations, note in diagram 1 that the ratio of the width of the shaded area, $ww/(n_p - 1)$, to the total area, we, is increasing and approaches 1 as n_p increases. This is equivalent to saying that s' increases and approaches 1 as n_p increases. Similarly, one can see that, as n_p increases the efficiency, which is the ratio of the white area to the total area, we is decreasing and approaching 0. On the other hand, based on diagram 2, one sees that s' (which is the ratio of the width of the shaded area to the total area) decreases as n increases, while the efficiency increases.

In conclusion, the dynamics of the fundamental parameters can be shown to provide a type of summary of the behavior of the parallel algorithm. Figure 12 contains the (approximate) level curves for the S, E, s, and s' parameters for the Cholesky algorithm implementation

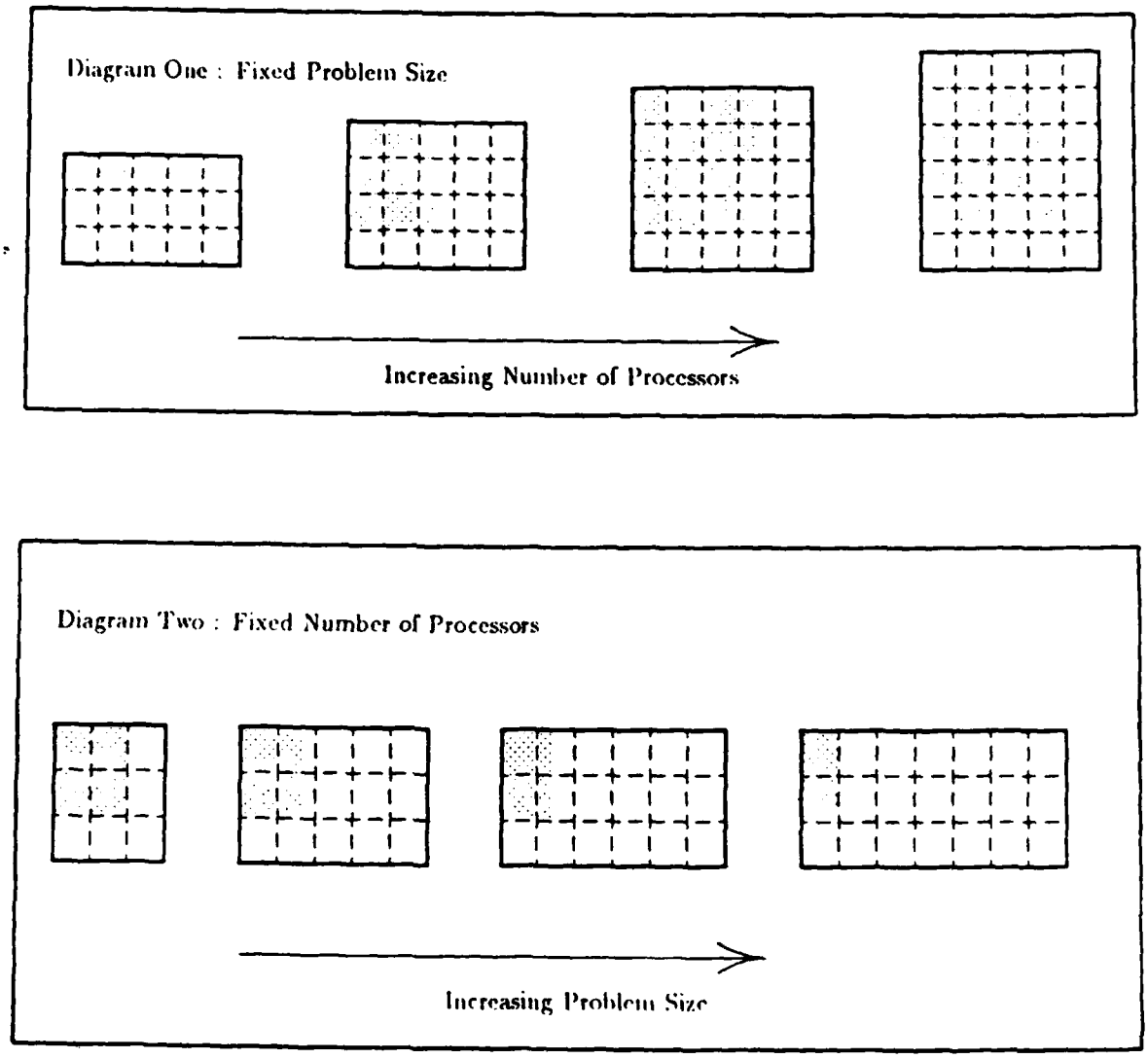
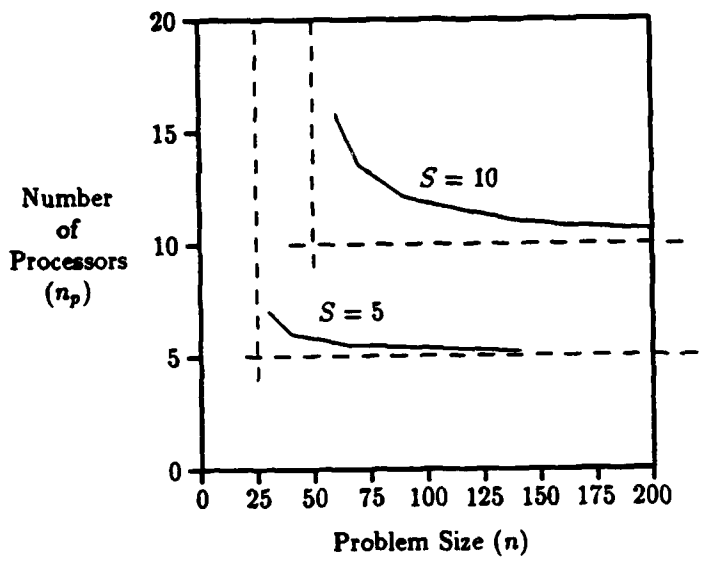
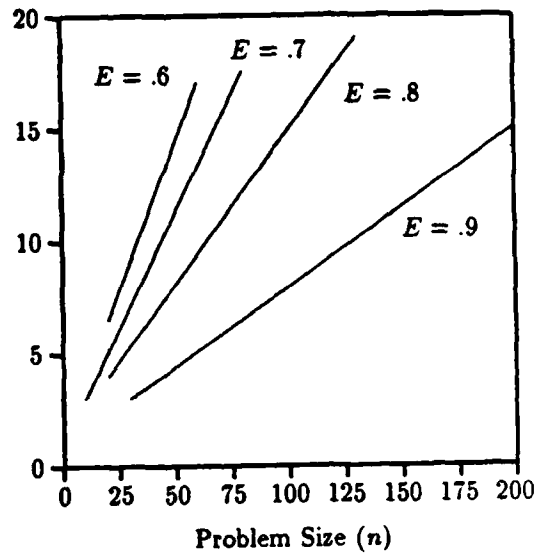


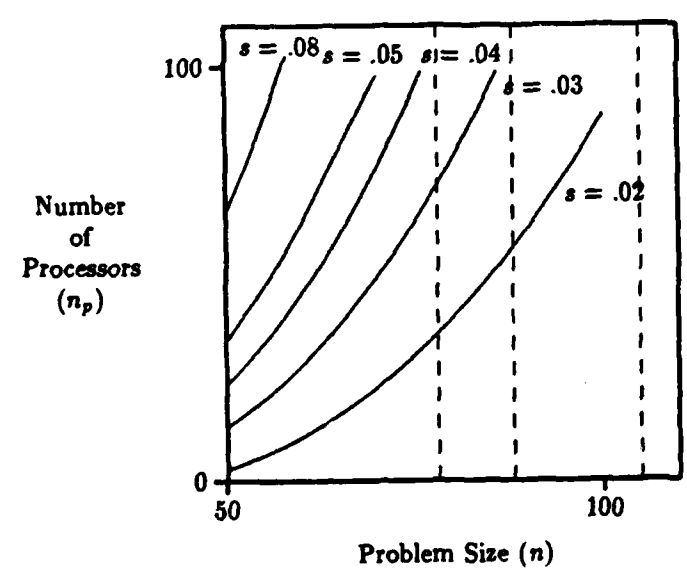
Figure 11. Boxed representations of parallel tasks.



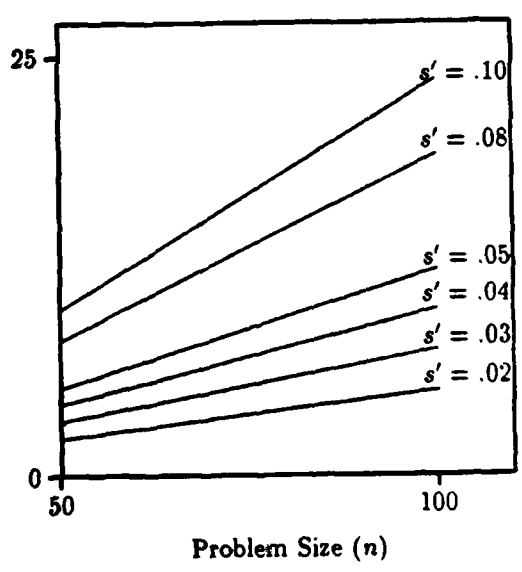
(a) Constant speedup curves.



(b) Constant efficiency curves.



(c) Constant s curves.



(d) Constant s' curves.

Figure 12. Level curves for S, E, s, and s'.

discussed in Section 3. Figure 12b shows that constant efficiency can be maintained by a linear choice of pairs (n, n_p) as $n, n_p \rightarrow \infty$. This choice of pairs will result in increasingly larger speedups, since each efficiency curve intersects the hyperbolic speedup curves in infinitely many points. Moreover, in Fig. 12a each line representing a constant value of n_p is a horizontal asymptote for the speedup curve given by $S = n_p$. Similarly, in Fig. 12c the lines representing constant n values are vertical asymptotes for the constant s curves and constant speedup curves. In Fig. 12d, a constant s' is also maintained by a linear combination of n and n_p. This statement is consistent with the findings reported in Ref. 2 that a constant s' value (and hence a linear speedup) can be obtained for certain problems by allowing n to increase linearly with n_p. Finally, the similarity of the level curves in Fig. 12b and 12d can be explained by noting that, as the number of processors increases, the difference between the efficiency E and the parallel fraction p' approaches zero $[\lim_{n_p \rightarrow \infty}(E - p') = 0\ ]$.

4. Conclusions

The widespread practice of measuring parallel performance solely in terms of constant entities s, p, independent of n and n_p, leads to a weak model which has very little predictive or descriptive value. Historically, this practice originated in the framework of vector processing, where the vectorizable portion of a loop constitutes a portion of work that can be executed with 100 percent efficiency, while the unvectorized portion constitutes work that must be performed serially (one operation at a time). This same reasoning, however, does not apply to parallel processing since, in general, tasks are composed of streams of operations. In this context, it is the sequential nature of the operations in a given stream that bounds the speedup attainable by an algorithm. Because independent streams may have different lengths, partitioning these for parallel execution is not as simple as dividing them equally among available processors (*i.e.*, p/n_p). The incongruent packing of these streams yields idle periods which contribute to the inefficiency of the system and result in the waiting behavior which characterizes the algorithm's performance.

Therefore, the work parameters ww, wa, and we introduced in this report are useful because they provide a natural means for interpreting the performance of a parallel algorithm for various problem and ensemble sizes in terms of its waiting behavior. These parameters also provide a connecting link between the overly simplistic "serial" and "parallel" fraction parameters and the very detailed timing statistics obtained empirically. Moreover, using these parameters to describe the equivalence of the alternate formulations of speedup introduces a new dynamic view of the "serial" and "parallel" fractions as variables depending on n and n_p. This new point of view permits the formulation of more sophisticated parallel performance models which accurately reflect both theoretical results and empirical observations.

4.1 Future Research

Plans for future work include three aspects. The first aspect involves the use of the new speedup and efficiency definitions to characterize parallel algorithms. The second aspect involves extending the formulations of speedup by introducing time as a variable and by replacing the "serial" and "parallel" fraction pairs (s, p) or (s', p') with probability distributions based on the number of processors. The third aspect involves extending the models to include properties of scaled speedup. The second aspect will permit the fundamental parameters to be treated from a statistical point of view. The concept of scaled speedup (introduced in Ref. 2 and Ref. 3) is important because of its connection to the behavior of speedup on various curves in the (n_p, n) plane.

References

1. Amdahl, G. M., "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities," *AFIPS Conference Proceedings*, Reston, Va., April 1967.
2. Gustafson, John L., Montry, Gary R., and Benner, Robert E., "Development of Parallel Methods for a 1024-Processor Hypercube," *SIAM Journal on Scientific and Statistical Computing*, 9(4), July 1988.
3. Womble, D. E., Allen, R. C., and Baca, L. S., "Invariant Embedding and the Method of Lines for Parallel Computers," Sandia National Laboratories, Albuquerque, N.M. 87185. Internal Report.
4. Fox, Geoffrey and Otto, Steve W., "Concurrent Computation and the Theory of Complex Systems," Heath, Michael T., editor, *Hypercube Multiprocessors, 1986*, pages 244–268, SIAM, 1986.
5. Forsythe, George and Moler, Cleve, *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1967.
6. Franklin, Joel N., *Matrix Theory*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 1968.
7. Cosnard, M., Marrakchi, M., and Robert, Y., "Parallel Gaussian Elimination on an MIMD Computer," *Parallel Computing*, 6(3), March 1988.
8. Dongarra, J. J., Gustafson, F. G., and Karp, A., "Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine," *SIAM Review*, 26(1), January 1984.
9. George, Alan, Heath, Michael, and Liu, Joseph, "Parallel Cholesky Factorization on a Shared Memory Multiprocessor," *Linear Algebra and Its Applications*, 77:165–187, 1986.

Appendix A. Dynamics of Fundamental Parameters

The following tables illustrate the behavior of the fundamental parameters for various problem sizes and ensemble sizes. They were computed based on tables from several sources as specified in the table captions.

Table A.1. Speedup, s, and s' — Example 1. Table compiled on the basis of Table 2, Section 5.5, p. 14 in Ref. 2.

	Number Of Processors				
Problem Size	4	16	64	256	1024
			Speedup		
9×2^{12}	3.98	15.86	62.00	226.395	639
9×2^{10}	3.96	15.52	57.036	167.225	
9×2^{8}	3.87	14.32	44.3		
9×2^{6}	3.60	11.07			
9×2^{4}	2.96				
			s Values		
9×2^{12}	0.00115	0.0006	0.00051	0.00052	0.00059
9×2^{10}	0.00301	0.0020	0.00194	0.00208	
9×2^{8}	0.01067	0.0077	0.00759		
9×2^{6}	0.03663	0.02967			
9×2^{4}	0.11708				
			s' Values		
9×2^{12}	0.00457	0.00946	0.03161	0.11688	0.37634
9×2^{10}	0.01192	0.03201	0.11034	0.34814	
9×2^{8}	0.04136	0.11163	0.32876		
9×2^{6}	0.13201	0.32852			
9×2^{4}	0.34659				

Table A.2. Speedup, s, and s' — Example 2. Table compiled on the basis of Table 5, Section 6.5, p. 23 in Ref. 2.

	Number Of Processors				
Problem Size	4	16	64	256	1024
			Speedup		
2×2^{12}	3.959	15.473	58.534	201.630	319.150
2×2^{10}	3.908	14.812	51.260	132.259	
2×2^{8}	3.780	13.136	34.091		
2×2^{6}	3.472	8.990			
2×2^{4}	2.578				
			s Values		
2×2^{12}	0.00341	0.00227	0.00148	0.00106	0.00093
2×2^{10}	0.00781	0.00535	0.00395	0.00367	
2×2^{8}	0.01942	0.01454	0.01393		
2×2^{6}	0.05069	0.05198			
2×2^{4}	0.18384				
			s' Values		
2×2^{12}	0.01352	0.03512	0.08676	0.21322	0.49350
2×2^{10}	0.03053	0.07920	0.20223	0.48426	
2×2^{8}	0.07339	0.19095	0.47475		
2×2^{6}	0.17600	0.46731			
2×2^{4}	0.47396				

Table A.3. Speedup, s, and s' — Example 3. Table compiled on the basis of Table 8, p. 29 in Ref. 2.

	Number Of Processors				
Problem Size	4	16	64	256	1024
			Speedup		
2×2^{11}	3.954	15.462	57.464	177.491	351.241
2×2^{9}	3.925	14.781	47.182	98.542	
2×2^{7}	3.805	12.599	28.088		
2×2^{5}	3.437	8.182			
2×2^{3}	2.561				
			s Values		
2×2^{11}	0.00387	0.00232	0.00181	0.00173	0.00187
2×2^{9}	0.00634	0.00550	0.00566	0.00627	
2×2^{7}	0.01710	0.01800	0.02029		
2×2^{5}	0.05460	0.06370			
2×2^{3}	0.18730				
			s' Values		
2×2^{11}	0.01529	0.03588	0.10375	0.30788	0.65763
2×2^{9}	0.02490	0.06123	0.26695	0.61748	
2×2^{7}	0.06507	0.22674	0.57003		
2×2^{5}	0.18765	0.52121			
2×2^{3}	0.47967				

Table A.4. Speedup, s, and s' — Example 4. Table compiled on the basis of Table 1, p. 8, Example 1 in Ref. 3.

	Number Of Processors				
Problem Size	4	16	64	256	1024
			Speedup		
2×2^{12}	2.606	10.007	32.468	61.040	64.936
2×2^{10}	2.580	8.849	19.513	21.139	
2×2^{8}	2.468	6.129	7.600		
2×2^{6}	2.087	2.824			
2×2^{4}	1.333				
			s Values		
2×2^{12}	0.17824	0.03993	0.01542	0.01233	0.01444
2×2^{10}	0.18353	0.05388	0.03619	0.04357	
2×2^{8}	0.20702	0.10737	0.11779		
2×2^{6}	0.30556	0.31111			
2×2^{4}	0.66667				
			s' Values		
2×2^{12}	0.46456	0.39956	0.50051	0.76455	0.93750
2×2^{10}	0.47365	0.47674	0.70615	0.92102	
2×2^{8}	0.51082	0.65806	0.89524		
2×2^{6}	0.63768	0.87843			
2×2^{4}	0.88889				

Table A.5. Speedup, s, and s' — Example 5. Table compiled on the basis of Table 1, p. 8, Example 2 in Ref. 3.

	Number Of Processors				
Problem Size	4	16	64	256	1024
			Speedup		
2×2^{12}	3.940	15.128	49.085	92.280	98.170
2×2^{10}	3.902	13.384	29.513	31.972	
2×2^{8}	3.727	9.258	11.480		
2×2^{6}	3.130	4.235			
2×2^{4}	2.111				
			s Values		
2×2^{12}	0.00506	0.00384	0.00482	0.00696	0.00922
2×2^{10}	0.00840	0.01303	0.01855	0.02748	
2×2^{8}	0.02439	0.04855	0.07262		
2×2^{6}	0.09259	0.18519			
2×2^{4}	0.29825				
			s' Values		
2×2^{12}	0.01993	0.05814	0.23674	0.64204	0.90501
2×2^{10}	0.03227	0.17447	0.54742	0.87854	
2×2^{8}	0.09091	0.44946	0.83365		
2×2^{6}	0.28986	0.78631			
2×2^{4}	0.62963				